Real Time Processing

Karthik Ramasamy
Streamlio
Information Age

Real-time is key

Enormous explosions leave a smoky ruin in Tianjin, China nyt.ms/1DC7jkX

Archeologists believe they have found a mass burial site of suspected plague victims cnn.it/196QqM

JACC finally confirms flaparon is from MH370 bit.ly/1KO7ea
Real Time Connected World

Internet of Things
30 B connected devices by 2020

Health Care

Machine Data
40% of digital universe by 2020

Connected Vehicles
Data transferred per vehicle per month
4 MB -> 5 GB

Digital Assistants (Predictive Analytics)
Siri/Cortana/Google Now

Augmented/Virtual Reality
$150B by 2020 [2]
Oculus/HoloLens/Magic Leap

Why Real Time?

REAL TIME TRENDS
Emerging break out trends in Twitter (in the form #hashtags)

REAL TIME CONVERSATIONS
Real time sports conversations related with a topic (recent goal or touchdown)

REAL TIME RECOMMENDATIONS
Real time product recommendations based on your behavior & profile

REAL TIME SEARCH
Real time search of tweets
Value of Data

It's contextual

Value of Data to Decision-Making

Time-critical Decisions

Information Half-Life In Decision-Making

Traditional “Batch” Business Intelligence

What is Real-Time?

It's contextual

> 1 HOUR
- high throughput
- adhoc queries
- monthly active users
- relevance for ads

10 MS – 1 SEC
- approximate
- ad impressions count
- hash tag trends

< 500 MS
- latency sensitive
- deterministic workflows
- fanout Tweets
- search for Tweets

< 1 MS
- low latency
- Financial Trading

BATCH

REAL TIME

OLTP

REAL REAL TIME
Real Time Analytics

STREAMING

Analyze data as it is being produced

INTERACTIVE

Store data and provide results instantly when a query is posed
Real Time Use Cases

Online Services
10s of ms
Transaction log, Queues, RPCs

Real Time
10-100s of ms
Change propagation, Streaming analytics

Data for Batch Analytics
secs to mins
Log aggregation, Client events
Real Time Stack

Components: Many moving parts

- Data Collectors
- Messaging
- Storage
- Compute
Scribe

Open source log aggregation
Originally from Facebook. Twitter made significant enhancements for real time event aggregation

High throughput and scale
Delivers 125M messages/min. Provides tight SLAs on data reliability

Runs on every machine
Simple, very reliable and efficiently uses memory and CPU
Event Bus & Distributed Log
Twitter Messaging

Core Business Logic (tweets, fanouts ...)

Deferred RPC | Gizzard | Database | Search

Kestrel | Kestrel | Kestrel | Book Keeper | My SQL | Kafka

Scribe

Kala

Deferred RPC

HDFS
Kestrel Queue

MESSAGE QUEUE

MULTIPLE PROTOCOLS

HIGHLY SCALABLE

LOOSELY ORDERED
Kestrel Queue
Kestrel Limitations

- Durability is hard to achieve
- Read-behind degrades performance
  Too many random I/Os
- Adding subscribers is expensive
- Scales poorly as #queues increase
- Cross DC replication
Kafka Queue

CLIENT STATE MANAGEMENT

CONSUMER SCALABILITY

HIGH THROUGHPUT

SIMPLE FOR OPERATIONS
Kafka Limitations

- Relies on file system page cache
- Performance degradation when subscribers fall behind - too much random I/O
- Scaling to many topics
- Loss of data
- No centralized operational stats
Rethinking Messaging

Unified Stack - tradeoffs for various workloads

Multi tenancy

Ease of Manageability

Durable writes, intra cluster and geo-replication

Scale resources independently

Cost efficiency
Event Bus - Pub-Sub

Event Bus + Distributed Log
Distributed Log

Publisher → Write Proxy → BK → Read Proxy → Subscriber

Metadata - ZK → Distributed Log
Distributed Log @Twitter

01 Manhattan Key Value Store
02 Durable Deferred RPC
03 Real Time Search Indexing
04 Pub Sub System
05 Globally Replicated Log
Distributed Log @Twitter

- 400 TB/Day IN
- 20 PB/Day OUT
- 2 Trillion Events/Day PROCESSED
- 5-10 MS latency
Twitter Heron

Next Generation Streaming Engine
Twitter Heron
Better Storm

- Container Based Architecture
- Separate Monitoring and Scheduling
- Simplified Execution Model
- Much Better Performance

"Can you redo this manuscript, John, and make it less stupid?"
Twitter Heron

Design: Goals

- Fully API compatible with Storm
  Directed acyclic graph
  Topologies, Spouts and Bolts

- Use of main stream languages
  C++, Java and Python

- Batching of tuples
  Amortizing the cost of transferring tuples

- Task isolation
  Ease of debug-ability/isolation/profiling

- Support for back pressure
  Topologies should self adjusting

- Efficiency
  Reduce resource consumption
Twitter Heron

- Guaranteed Message Passing
- Horizontal Scalability
- Robust Fault Tolerance
- Concise Code-Focus on Logic
Heron Terminology

**Topology**
Directed acyclic graph
vertices = computation, and
edges = streams of data tuples

**Spouts**
Sources of data tuples for the topology
Examples - Kafka/Kestrel/MySQL/Postgres

**Bolts**
Process incoming tuples, and emit outgoing tuples
Examples - filtering/aggregation/join/any function
Heron Topology

Spout 1 → Bolt 1 → Bolt 4
Spout 2 → Bolt 2 → Bolt 5
Spout 2 → Bolt 2 → Bolt 3
Stream Groupings

01 Shuffle Grouping
Random distribution of tuples

02 Fields Grouping
Group tuples by a field or multiple fields

03 All Grouping
Replicates tuples to all tasks

04 Global Grouping
Send the entire stream to one task
Heron

Architecture: High Level

Scheduler

Topology Submission

Topology 1

Topology 2

Topology N
Heron

Architecture: Topology

Topology Master

Sync Physical Plan

ZK Cluster

Logical Plan, Physical Plan and Execution State

Metrics Manager

Stream Manager

CONTAINER

I1 I2 I3 I4

CONTAINER

Stream Manager

Metrics Manager

I1 I2 I3 I4
Heron
Stream Manager: BackPressure
Stream Manager

Stream Manager: BackPressure
Heron

Stream Manager: Spout BackPressure
Heron Use Cases

- Realtime ETL
- Real Time BI
- Spam Detection
- Real Time Trends
- Realtime ML
- Real Time Ops
Heron
Sample Topologies
Heron @Twitter

Heron has been in production for 3 years

1 stage vs 10 stages

3x reduction in cores and memory
Heron

Performance: Settings

<table>
<thead>
<tr>
<th>COMPONENTS</th>
<th>EXPT #1</th>
<th>EXPT #2</th>
<th>EXPT #3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spout</td>
<td>25</td>
<td>100</td>
<td>200</td>
</tr>
<tr>
<td>Bolt</td>
<td>25</td>
<td>100</td>
<td>200</td>
</tr>
<tr>
<td># Heron containers</td>
<td>25</td>
<td>100</td>
<td>200</td>
</tr>
</tbody>
</table>
Heron
Performance: Atmost Once

Throughput

CPU usage

<table>
<thead>
<tr>
<th>Spout Parallelism</th>
<th>Heron (paper)</th>
<th>Heron (master)</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>249</td>
<td>1,545</td>
</tr>
<tr>
<td>100</td>
<td>965</td>
<td>1,920</td>
</tr>
<tr>
<td>200</td>
<td>1,920</td>
<td>10,200</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Spout Parallelism</th>
<th># cores used (Heron)</th>
<th># cores used (master)</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>32</td>
<td>54</td>
</tr>
<tr>
<td>100</td>
<td>137</td>
<td>217.5</td>
</tr>
<tr>
<td>200</td>
<td>261</td>
<td>397.5</td>
</tr>
</tbody>
</table>

5 - 6x

1.4 - 1.6x
Heron

Performance: CPU Usage

**Heron (paper)**

**Heron (master)**

4-5x
Heron @Twitter

- > 400 Real Time Jobs
- 500 Billions Events/Day PROCESSED
- 25-50 MS latency
Stateful Processing in Heron

- Optimistic Approaches
- Pessimistic Approaches
Tying Together
Lambda Architecture
Combining batch and real time

New Data

Client

Heron

Hadoop
Lambda Architecture - The Good

Scribe Collection Pipeline → Event Bus → Heron Compute Pipeline → Results
Lambda Architecture - The Bad

- Have to write everything twice!
- Subtle differences in semantics
- Have to fix everything (may be)
- How much Duct Tape required?
- What about Graphs, ML, SQL, etc?
Summingbird to the Rescue

Heron Topology

Message broker

Summingbird Program

HDFS

Scalding/Map Reduce

Online key value result store

Batch key value result store

Client
Twitter Heron: Stream Processing at Scale

Sanjeev Kulkarni, Nikunj Bhagat, Maosong Fu, Vikas Kedigehalli, Christopher Kellogg, Sailes Math, Jignesh M. Patel, Karthik Ramasamy, Siddartha Taneja

Twitter, Inc., *University of Wisconsin – Madison

Storm @Twitter

Ankit Toshniwal, Siddharth Taneja, Amit Shukla, Karthik Ramasamy, Jignesh M. Patel*, Sanjeev Kulkarni, Jason Jackson, Krishna Gade, Maosong Fu, Jake Donham, Nikunj Bhagat, Sailesh Mittal, Dmitri Ryabay

Twitter, Inc., *University of Wisconsin – Madison

Streaming@Twitter

Maosong Fu, Sailesh Mittal, Vikas Kedigehalli, Karthik Ramasamy, Michael Barry, Andrew Jorgensen, Christopher Kellogg, Neng Lu, Bill Graham, Jingwei Wu

Twitter, Inc.

Twitter Heron: Towards Extensible Streaming Engines

Karthik Ramasamy†, Ashvin Agrawal†, Avrilia Floratou †, Maosong Fu*, Bill Graham†, Andrew Jorgensen†

Mark Li†, Neng Lu†, Sriram Rao†, Cong Wang†

Twitter, Inc. { @karthikz, @Louis_Fumaosong, @hilgraham, @ajorgensen, @objcione, @hneng90, @cOngwang}†

Microsoft {asagr, avflor, strramra}@microsoft.com
Curious to Learn More?

DistributedLog: A high performance replicated log service

Sijie Guo
sijie@twitter.com
Twitter

Robin Dhamankar
robin.dhamankar@gmail.com
Twitter

Leigh Stewart
lstewart@twitter.com
Twitter

Dhalion: Self-Regulating Stream Processing in Heron

Avrilila Floratou
Microsoft
avflor@microsoft.com

Ashvin Agrawal
Microsoft
asagr@microsoft.com

Bill Graham
Twitter, Inc.
@billgraham

Sriram Rao
Microsoft
sriramra@microsoft.com

Karthik Ramasamy
Twitter, Inc.
@karthikz
Interested in Heron?

HERON IS OPEN SOURCED
CONTRIBUTIONS ARE WELCOME!

https://github.com/twitter/heron

http://heronstreaming.io

FOLLOW US @HERONSTREAMING
Interested in Distributed Log?

DISTRIBUTED LOG IS OPEN SOURCED

CONTRIBUTIONS ARE WELCOME!

https://github.com/twitter/heron

http://distributedlog.io

FOLLOW US @DISTRIBUTEDLOG
We are hiring @Streamlio

SOFTWARE ENGINEERS
PRODUCT ENGINEERS
SYSTEM ENGINEERS
UI ENGINEERS/UX DESIGNERS

EMAIL CAREERS [AT] STREAML.IO
Any Question ???

WHAT  WHY  WHERE  WHEN  WHO  HOW
Get in Touch
@karthikz
THANKS FOR ATTENDING !!!